Modeling images of natural 3D surfaces: overview and potential applications

***Abstract*— Generative models of natural images have long been used in computer vision. However, since they only describe the statistics of 2D scenes, they fail to capture all the properties of the underlying 3D world. Even though such models are sufficient for many vision tasks, a 3D scene model is needed when it comes to inferring a 3D object or its characteristics. In this paper, we present such a generative model, incorporating both a multiscale surface prior model for surface geometry and reflectance, and an image formation process model based on realistic rendering, that accounts for the physics of image generation. We focus on the computation of the posterior model parameter densities, and on the critical aspects of the rendering. We also discuss how to efficiently invert the model within a Bayesian framework. We present a few potential applications, such as asteroid modeling and planetary topography recovery, illustrated by promising results on real images.**

I. Introduction

The model we study in this paper is intended to describe 3D natural surfaces such as planetary or asteroid relief, as well as optical images of these surfaces, taken under different viewing and lighting conditions.

Natural image statistics can be efficiently described by 2D models, as shown in various studies such as [1]–[5]. These image models are mostly bidimensional, and they capture some of the characteristics of natural objects, such as scale invariance, spatial adaptivity and various roughness or regularity properties. Within a Bayesian framework [6], they can be used to infer the model parameters from an observation (or a set of observations), thus providing estimates of the modeled characteristics.

However, an image is not a simple representation of a natural 3D object, it is in fact a measurement, corrupted by blur and noise, of a 2D *rendering* of such an object. Therefore it is not appropriate, in general, to model an image directly as a natural phenomenon, and there is usually no simple correspondence between the inferred model parameters and the surface parameters (the former are usually a complex mixture of the latter), except in some simple cases [7]. Therefore the imaging model should be taken into account. Furthermore, the object model should relate to the physical properties of the studied surface, such as shape, reflectance, roughness etc.

We propose to build a full generative model that combines a 3D surface model with a realistic imaging process to describe both the scene and the various observations of this scene. This model is described in Section II; the surface model includes topography, reflectance and various hyperpriors, whereas the imaging model consists of an accurate rendering algorithm followed by a degradation process. Our contribution consists of both the surface model and the rendering technique. However, the main contribution of this paper is to derive a complete generative model for images of natural surfaces. After describing the forward problem, in Section III we detail a few potential applications to computer vision using Bayesian inference, show preliminary results and discuss the related challenges.

II. Generative model

We first define a surface model $\mathbf{S}$ which consists of a set of 3D vertices $\mathbf{v}$ (geometry) forming a triangular mesh, and scalar albedos ρ, one for each triangle. We assume that all the parameters are random variables governed by a joint probability distribution. The geometry model is described in Section A. It comprises a set of coefficients $\mathbf{w}$ (wavelet transform of $\mathbf{v}$) conditioned upon the roughness parameters λ, γ and q. The reflectance model is described in Section B and is made of coefficients ω (wavelet transform of ρ) of roughness parameters ε; we also define a model map m and scattering parameters κ. The camera and light parameters are denoted by $\mathbf{\Theta}$. An image I is obtained from $\mathbf{S}$ and $\mathbf{\Theta}$ by rendering, as explained in Section C. Any observed image X depends on I and $\mathbf{\Theta}$ through a degradation model given in Section D. The relationships between all these variables are given as a graphical model in Fig. 1, where each arrow represents a conditional density, and each leaf node a prior density.

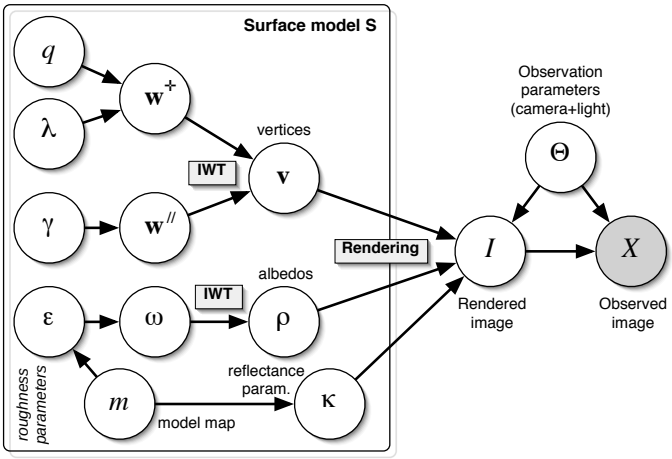

Fig. 1. Graphical model, or hierarchy of the random variables in the proposed generative model, including **rendering** and **inverse wavelet transform** (IWT).

A. Multiscale surface geometry model

Fractals have long been used to synthesize realistic looking planetary terrains, because of their resemblance to natural objects [1], [8]. From a qualitative point of view, they certainly

exhibit similar statistical properties, such as scale invariance. We propose to derive a multiscale roughness model that accounts for these properties, by building an appropriate probability density function of the vertex variables $\mathbf{v}$.

In this paper, we are interested in modeling any kind of surfaces, such as asteroids or entire planets, which are topologically different from flat open landscapes commonly used in terrain simulation. For this purpose, we choose to use a subdivided mesh [9] as the topological support of $\mathbf{v}$. On each site of this support lies a 3D vertex variable. This support is semi-regular, since we start from a root mesh of fixed connectivity (such as a hexagonal grid in the planar case, or an icosahedron in the spherical case), then we subdivide it regularly by recursively adding a vertex between each pair of existing vertices (see Fig. 2).

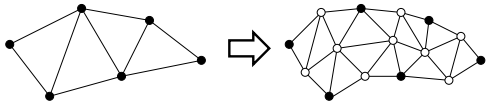

Fig. 2. Subdivision scheme used to produce a finer mesh from an existing triangular mesh: a new vertex (white) is added between each pair of 2 vertices (black), using a prediction or interpolation rule.

A possible way of studying fractals is to look for statistical self-similarities. Simple probabilistic estimators can be used instead of looking for repetition and scaling of particular geometrical shapes. If there is a scale invariant probability function fitting to the data, the object is said to be statistically scale invariant, and we can call it fractal. Usually a spectral representation such as the Fourier transform (when available) gives access to the distribution of the average size of object features as a function of the scale, regardless of the location. For perfect spherical objects, spherical harmonics provide a powerful spectrum analysis tool. However, the surfaces we model have an irregular sampling in general, since the radius variations are large w.r.t. the object radius, therefore we prefer to use a more flexible tool such as wavelets in order to access the scale of geometric features.

1) Wavelet transform of a surface: Now that the topology is properly defined, how do we deal with the 3D geometry? The key point of this subdivision scheme is the new vertex prediction, which is achieved by interpolation. The simplest is to take the midpoint of the edge, but leads to an unwanted piecewise planar surface. Therefore we prefer using a smooth scheme [10], involving 8 parents for each new vertex instead of 2. We use this scheme in the regular case (both edge vertices have 6 neighbors), otherwise we use another one [11]. If $V(\mathbf{v}_j)$ denotes the 8 neighbors of a new vertex $\mathbf{v}_j$, the prediction function is denoted by $\mathbf{B}_j\left(V(\mathbf{v}_j)\right)$.

A subdivided mesh at level J is given. The basic idea is to split the sites into two interleaved sets: the topological midpoints and their closest parents (respectively white and black points on Fig. 2). Then, the former are predicted from the latter using $\mathbf{B}_j$. The difference between actual and predicted vertices gives us the *wavelet details*, since it represents at each level the difference between a smooth approximation and the actual surface. These details are topologically located at the same "white" sites as the midpoints, of index j:

$$\mathbf{w}_j = \mathbf{v}_j - \mathbf{B}_j\big(V(\mathbf{v}_j)\big) \tag{1}$$

Without lifting, the wavelet functions would not have sufficient smoothness properties, such as spectral selectivity, needed to capture scale properties of natural surfaces. Therefore we use the lifting scheme [12], consisting of adding to each "black" vertex (index k) a linear combination of the nearest wavelet details at the "white" sites (index j). We choose $\tau = 3/4$.

$$\mathbf{v}_k \leftarrow \mathbf{v}_k + \tau \sum_{j \sim k} \mathbf{w}_j \tag{2}$$

Finally, the wavelet transform is performed by recursively applying Eq. (1) to the "white" sites then the lifting at the "black" sites, in the reverse order from the subdivision, N times. The result is N levels of details, plus one coarse approximation of the mesh at subdivision level $J - N$. It is simple to invert, starting with the lifting step and replacing the addition by a subtraction in Eq. (2), then inverting Eq. (1) which consists of predicting the midpoints by using B_j and adding the coefficients $\mathbf{w}_j$. Filtering the detail coefficients provides a simple mesh smoothing technique [13].

2) Local scale and direction: The wavelet functions are actually defined in a topological space, which is semi-regular, and do not reflect the local geometry of the studied object. Thus, the coefficients encode absolute variations of the geometry between two approximation levels, regardless of the size of the triangles in the mesh. However, a given variation does not have the same physical meaning for different point densities. To account for that, we define the notion of local scale. This scale has nothing to do with the (integer) levels of the transform: at a given level there is a mixing of various scales depending on the local mesh density. The local scale for $\mathbf{v}_j$ is defined so that we can account for local deformation of each triangle (see Fig. 3). L is the length of the edge $\mathbf{v}_a\mathbf{v}_b$ in the approximation mesh (j is the midpoint of (a, b)), l is a distance from $\mathbf{v}_j$ to a parent of order 2, and the angle α encodes the skew of the triangle. The scale is actually an average of the scales of both triangles sharing the same edge.

$$s_j = L\left(\frac{3}{4}\frac{L^2}{l^2}(\cos\alpha + \sin\alpha)^2 + 4\right)^{-1/2} \tag{3}$$

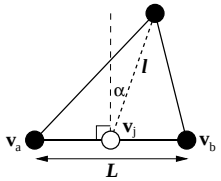

Fig. 3. Local deformation of the mesh around a wavelet coefficient as the midpoint of the edge a_1a_2: definition of the lengths L and l, skew angle α.

Like the approximation coefficients, the wavelet details are 3D vectors. The former have an obvious meaning, i.e. the same object at a coarser resolution, whereas the latter embed details both along and orthogonal to the surface. To provide a really useful transform, we have to separate these two components,

respectively the real geometric details (variations normal to the surface, denoted $\mathbf{w}_j^{\perp}$) and the surface sampling irregularities (variations parallel to the surface denoted $\mathbf{w}_j^{\parallel}$).

3) The surface model: Using wavelets on meshes we can perform the multiresolution analysis [14] of a surface for any topology, defined on a subdivided mesh. We have used such a representation of the asteroid 433 Eros; the geometry was given by the NEAR mission [15]. This way we have checked that this object is statistically scale-invariant. As shown in Fig. 4, the amplitude spectrum A, estimated by the spatial average of the amplitude of the geometric details $< |\mathbf{w}_j^{\perp}| >$, can be modeled by a scale invariant law where A_0 is a constant:

$$\log < |\mathbf{w}_j^{\perp}| > \simeq \; q \, \log(s_j) + \log A_0 \tag{4}$$

Here, the local scale s is related to the local spatial frequency f by $s = 1/f$. The scale invariance implies $A(f) = A_0 \, f^{-q}$, which describes the so-called "1/f" noise, a widely used model for natural objects [1], [3].

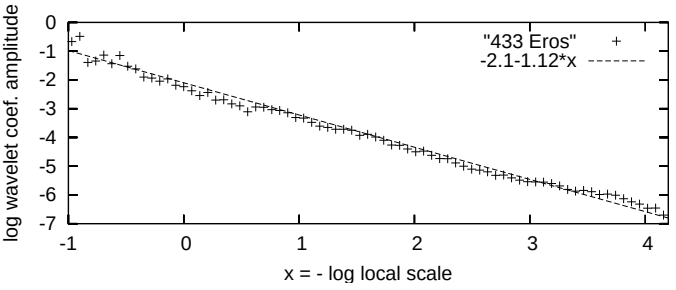

Fig. 4. Log-log plot representing the average size of the wavelet details of the asteroid 433 Eros as a function of the local scale, illustrating the statistically scale-invariant behavior of the surface.

This can be seen as a probabilistic model of the wavelet coefficients. It is closely related to a fractional Brownian motion [16], used to describe natural images. We extend this kind of model to natural surfaces (see Fig. 5). This wavelet transform, like more traditional 2D wavelet transforms, helps decorrelate the vertex random variables since the surface exhibits a self-similar behavior. Moreover, it conserves the number of coefficients since it is critically sampled, therefore we can reasonably assume that each normal wavelet detail coefficient can be accurately modeled as an independent random variable. To simplify, we use a zero-mean Gaussian. We build the joint distribution according to Eq. (4), and we define λ_j as local roughness parameters:

$$P\left(\mathbf{w}^{\perp} \,|\, \lambda,\, q\right) \propto \exp\left(-\sum_j \lambda_j \, s_j^{-2q} \left|\mathbf{w}_j^{\perp}\right|^2\right) \tag{5}$$

Thus, we construct a *spatially adaptive* fractal model applicable to a broad range of natural surfaces, whose properties are generally spatially varying.

On the other hand, the parallel coefficients $\mathbf{w}^{\parallel}$ are related to the smoothness of the surface sampling and their value should not have any influence on the actual object shape. A model involving them can be considered as a sampling regularity prior, whereas the model of Eq. (5) acts as a surface smoothness prior. We define a simple uniform prior on the scaled coefficients $\mathbf{w}_j^{\parallel}/s_j$ with the smoothness parameter γ:

$$P\left(\mathbf{w}^{\parallel} \,|\, \gamma\right) \propto \exp\left(-\gamma \sum_j \left|\mathbf{w}_j^{\parallel}/s_j\right|^2\right) \tag{6}$$

We express the prior distribution of the vertices $P(\mathbf{v})$ in the wavelet domain instead of the vertex domain by $P(\mathbf{w})$, and the conditional independence leads to $P(\mathbf{w}) = P(\mathbf{w}^{\perp})P(\mathbf{w}^{\parallel})$. Then the vertices $\mathbf{v}$ are obtained by inverse transform of $\mathbf{w}$.

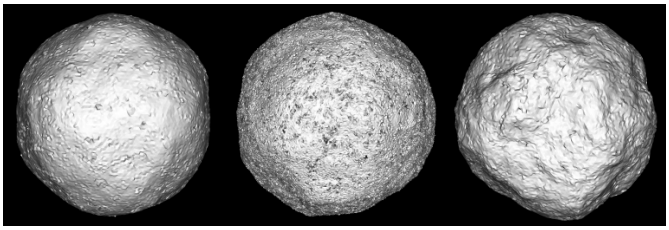

Fig. 5. Surfaces generated from the fractal model with $q = 1.1$ and uniform roughness λ (left: λ=0.5, middle: λ=1.5, right: λ=5). We have used the renderer described in Section II.C. with identical camera and light directions.

B. Hierarchical reflectance model

Images of natural surfaces are the product of albedo and shading. We propose to model the albedo field using existing natural image models [17], [18], which capture both the scale invariance and the spatial adaptivity via a multiresolution approach. The shading is modeled through a reflectance function f that depends on the surface geometry (vertices $\mathbf{v}$) and the observation parameters $\mathbf{\Theta}$.

To be more realistic, a model map should be included to account for the spatial variability of the terrain in real-world surfaces. This map consists of a discrete random variable m_j for each vertex of index j, and represents a local class of terrain (such as rock type, water, forest...); each class relates to a different albedo and reflectance function. More precisely, for each value of m, we have a multiscale albedo model of prior parameters ε, governed by the conditional distribution $P(\varepsilon \,|\, m)$, and a parametric reflectance function f^{κ} with the corresponding conditional parameter density $P(\kappa \,|\, m)$.

Thus we define a hierarchical model as follows: there is a prior distribution of the classes denoted by $P(m)$, then we have conditional densities $P(\varepsilon \,|\, m)$ and $P(\kappa \,|\, m)$, then the albedo model $P(\rho \,|\, \epsilon)$ and the reflectance model f^{κ}.

Let us focus on the albedo density. We can derive a multiscale model based on wavelets on a mesh, inspired from the geometry model. To ensure the physical constraints on the albedo, let us first define a modified albedo $\tilde{\rho} \in \mathbb{R}$ such that $\rho = \mu(\tilde{\rho})$ and $\rho \in [0, 1]$. We choose $\mu(x) = (1 + e^{-x})^{-1}$ which is a bijective sigmoid, so that we can easily use the density of $\tilde{\rho}$ instead of the density of ρ. Therefore, we need to use the Jacobian μ':

$$P(\rho \,|\, \varepsilon) = P(\tilde{\rho} \,|\, \varepsilon) \, \mu'(\tilde{\rho}) \tag{7}$$

We propose to use the same analysis scheme as in the previous section to derive albedo wavelets. On the mesh, we can first

define the albedos on the same topological sites as the vertices, then we get one albedo per triangle by averaging over the 3 triangle vertices. This way it is straightforward to apply Eqs. (1) and (2) using the scalar albedos instead of 3D vertices, in the prediction scheme as well as in the lifting step. To make a physical interpretation of the wavelet coefficients, we keep the same local scale estimate s_j related to the local geometry of the mesh. We denote the scalar albedo wavelet details by ω_j; their density is then given by:

$$P(\omega \,|\, \varepsilon) \propto \exp\Big(-\sum_j \varepsilon_j \ (\omega_j / s_j)^2 \Big) \tag{8}$$

The final albedos are thus obtained from the coefficients above by inverting the wavelet transform, averaging to get one albedo per triangle $\tilde{\rho}^{\triangle}$, then remapping into $[0,1]$ by the function μ.

C. Accurate rendering with derivatives

We need to produce an image from a fixed surface model $\mathbf{S}$ and a set of camera and light parameters $\mathbf{\Theta}$ (this is called *rendering* in computer graphics). We assume a pinhole camera model, which is a simple way to perform perspective projection, and for the light both a point source at infinity and an ambient component. $\mathbf{\Theta}$ contains both internal camera parameters (such as pixel size and focal length) as well as external parameters (position and orientation) and light parameters (direction and intensity). The major challenge is to compute accurate images, as well as their derivatives, i.e. how pixel intensities vary with changes to the surface and the observation parameters. The derivatives are required to perform the reconstruction via any gradient-based deterministic optimization algorithm.

We denote by $I_p(\mathbf{S},\, \mathbf{\Theta})$ the rendered intensity for the pixel p. This intensity is a product between surface albedo and local shading, which depends on geometry, lighting conditions, reflectance functions, and camera position and orientation. Computing accurate pixel intensities requires working in the object space, which means performing visibility determination for each pixel using computational geometry. This is the only way of obtaining an image that precisely corresponds to a 3D model, which is critical in some cases (see Section III-B).

1) Discrete intensity computation: When there are no occlusions or shadows, the contribution of a triangle $\triangle$ to a pixel p is the area of the triangle/pixel intersection polygon $A_p^{\triangle}$, times the irradiance $L^{\triangle}$. We denote this contribution by the fractional power $W_p^{\triangle}$:

$$W_p^{\triangle} = A_p^{\triangle}\, L^{\triangle} \tag{9}$$

When there are occlusions, the polygon is processed for hidden surface removal, as explained in paragraph 2 and Fig. 8.

Here the irradiance is assumed to be piecewise constant. We could use a more accurate piecewise linear model (cf. Phong model [19]), as it will be discussed in Section III.C.

The total intensity for pixel p is obtained by summing the fractional powers over the triangles intersecting the pixel p, i.e. the set $V(p)$, then multiplying them by a space-varying factor K_p encoding the exposure time and various transmission factors, as well as a geometric attenuation factor.

$$I_p = K_p \sum_{\triangle \in V(p)} W_p^{\triangle} \tag{10}$$

The irradiance is given by multiplying the direct light source intensity $I^\star$, the albedo ρ and the shading function f, which depends on the surface orientation (triangle normal $\mathbf{n}^{\triangle}$), and the camera and lighting directions $\mathbf{u}^c$ and $\mathbf{u}^\star$, through the incident angles $\theta_i^{\triangle}, \phi_i^{\triangle}$ and the viewing angles $\theta_r^{\triangle}, \phi_r^{\triangle}$ as defined on Fig. 6. There is also a non-directed light of intensity I^0 that accounts for ambient light and interreflections. The corresponding shading function is denoted f^0 and only depends on the triangle normal and the viewing direction.

$$L^{\triangle} = \rho^{\triangle} \ \Big(I^0\, f^0(\theta_r^{\triangle}, \phi_r^{\triangle}) + I^\star\, f(\theta_i^{\triangle}, \phi_i^{\triangle}, \theta_r^{\triangle}, \phi_r^{\triangle}) \Big) \tag{11}$$

In the following we assume a Lambertian model, but this can easily be extended to more realistic parametric reflectance functions as proposed by Oren & Nayar in [20]. Then we simply have $f = \cos\theta_i^{\triangle} = \mathbf{n}^{\triangle} \cdot \mathbf{u}^\star$ and f^0 becomes a constant:

$$L^{\triangle} = \rho^{\triangle} \ \big(I^0\, f^0 + I^\star\, \mathbf{n}^{\triangle} \cdot \mathbf{u}^\star \big) \tag{12}$$

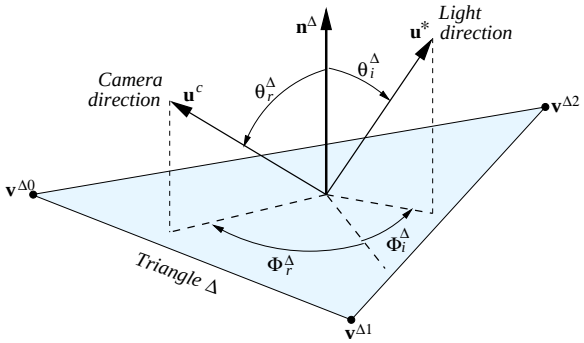

Fig. 6. Illustration of the light and camera directions and corresponding angles for a triangle $\triangle$ of the 3D surface.

2) Accurate visibility determination: We have developed a fast pixel integration method that computes, for each pixel p, the exact visible areas $A_p^{\triangle}$ of all the projected triangles $\triangle$ that overlap this pixel. It is an object-precision technique, since the size of the pixels does not affect the accuracy. This is made possible by combining bucket sorting (to build a list of triangles for each pixel) and depth buffers (to quickly reject totally hidden triangles), and by restricting the complex computations to the pixels where they are really required. This is used for both hidden surfaces and shadows, since shadows are surfaces hidden from the light source.

We notice that partial triangle occlusions occur only along curves (projected ridge lines), thus dramatically reducing the number of pixels of the image that require complex geometric computations to perform hidden surface removal. Outside ridge lines, triangles are either fully visible or fully hidden.

First, we determine the occlusion map, by rasterizing each edge of the mesh that defines a ridge, as follows. The normal to a triangle is used to test whether the triangle is front-facing

($\mathbf{n}^{\triangle} \cdot \mathbf{u}^{\star} > 0$) or back-facing. A ridge segment is defined as an edge separating a front-facing from a back-facing triangle, such that the front-facing one is the closest to the camera (otherwise it is a valley), see Fig. 7. The occlusion map is defined by the set of pixels that intersect the ridge segments.

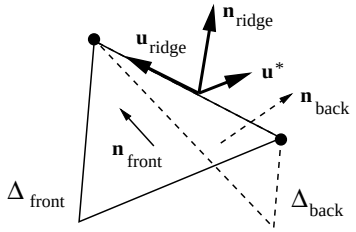

Fig. 7. Let $\mathbf{u}_{\text{ridge}}$ be the oriented edge (according to the front-face triangle) and the ridge normal the average $\mathbf{n}_{\text{ridge}} = (\mathbf{n}_{\text{front}} + \mathbf{n}_{\text{back}})/2$. The front-facing triangle $\triangle_{\text{front}}$ is *closest* to the camera $\Leftrightarrow |\mathbf{u}_{\text{ridge}}, \mathbf{u}^{\star}, \mathbf{n}_{\text{ridge}}| > 0$.

Then, we perform occluded surfaces removal only for the pixels in the occlusion map, which typically represent less than 1% of the image.

The principle of visible surface determination relies on recursively subtracting all the triangles that are in front of a particular triangle, thus obtaining a polygon, as shown in Fig. 8. Such methods exist in computer graphics [21], [22] but they do not perform recursive triangle subtraction using large triangular meshes. In addition to the geometry of line segments we use the topological connectivity of the mesh to design an algorithm that is robust to vertex and edge alignments occurring when intersecting adjacent triangles with a polygon. The polygon areas involved in the fractional intensity computation have to be determined, as well as their derivatives. We keep track of the original mesh vertices that generate all the geometric intersections involved in the subtraction algorithm (white vertices on Fig. 8). This way, it is possible to compute the intensity derivative w.r.t. any vertex, even in the case of complex occultations involving many vertices.

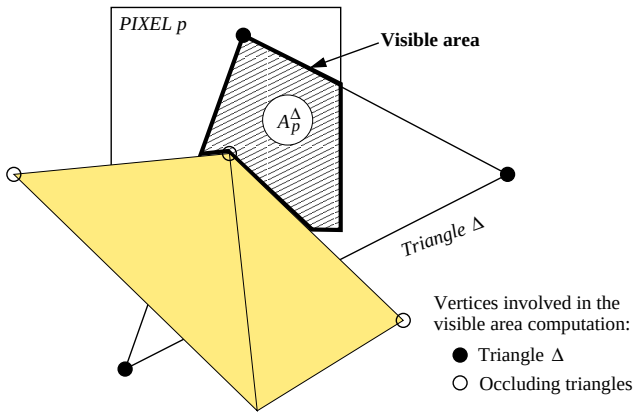

Fig. 8. For a pixel p, illustration of the visible part of the triangle $\triangle$ (dashed polygon), of area $A_p^{\triangle}$. It is obtained by subtracting all occluding triangles from the triangle/pixel intersection polygon.

3) Computing shadows: Shadow boundaries carry very important information on the 3D, independent of any albedo or reflectance estimation errors. Therefore they have to be taken into account accurately, at sub-pixel level. This is made possible by reusing the technique described above.

First, an orthographic projection along the light direction is used to determine, for each triangle, the list of occluding triangles. Then, for each occluded triangle, the area not in shadow is determined by subtracting the occluding triangles, projected using a different projection this time (along the light direction, but onto the shadowed triangle).

It is also possible to compute shadows at a triangle level, by determining for each triangle the area visible from the directed light source (as we do from the camera for hidden surface removal). Then we define a shadow rate for each triangle as the visible to total area ratio, which multiplies the second term in Eq. (12), so that triangles in full shadow will only receive ambient light. This approximation gives very good results when the triangles are small.

4) Computing the derivatives: The knowledge of the derivatives of the image intensity w.r.t. any parameter of the generative model, such as the surface or the camera and light parameters, is highly valuable. First, efficient *deterministic optimization* techniques require derivatives to estimate model parameters from observed data. Second, they can help compute the *uncertainty* of the generated models by providing a Gaussian approximation of the model probability distribution. Moreover, the intensity derivatives can be used to compute the optical flow related to changes in the vertices or camera parameters, thus enabling us to add *motion blur* to the rendering scheme.

The basic idea of the computation is the chain rule. All we need to know is the derivative of any function w.r.t. any variables this function directly depends on: for instance, the projection of a vertex only depends on the camera parameters and the 3D vertex; a fractional area $A_p^{\triangle}$ only depends on the projected vertices of $\triangle$ and the occluding triangles. Let us denote by $\mathbf{U}$ and $\mathbf{V}$ arbitrary vectors (such as vertices, albedos or areas). If we assume that we have n vectors $\mathbf{Z}_i$ functions of $\mathbf{U}$, and that $\mathbf{V}$ is a function of all $\mathbf{Z}_i$, then the corresponding derivatives are multiplied according to the chain rule to obtain the derivative of $\mathbf{V}$ w.r.t. $\mathbf{U}$:

$$\left[\frac{\partial \mathbf{V}}{\partial \mathbf{U}}\right] = \sum_{i=1}^{n} \left[\frac{\partial \mathbf{V}}{\partial \mathbf{Z}_i}\right] \left[\frac{\partial \mathbf{Z}_i}{\partial \mathbf{U}}\right] \tag{13}$$

This can be extended to a full derivative tree, encoding to the hierarchical relations between all variables in the rendering procedure, from $(\mathbf{S}, \boldsymbol{\Theta})$ to the intensities I_p.

D. Observed image formation model

In principle, the observed image is formed in 3 steps: 1) the projection onto the image plane, which produces a piecewise constant image since we assumed that the irradiance is constant over triangles; 2) the convolution by the point spread function (PSF) of the instrument; 3) the integration over each pixel. However, an equivalent model consists of replacing steps 2 and 3 by the convolution with a global PSF including both instrument and pixel PSF, then point sampling on a rectangular grid.

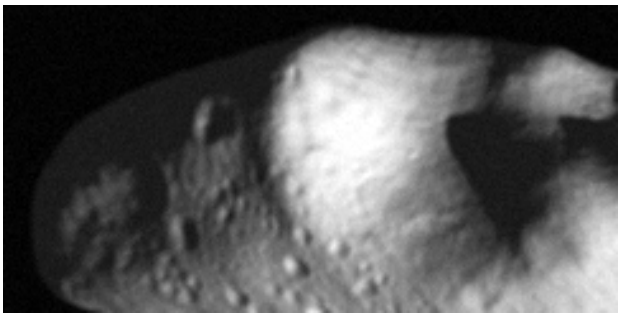

Fig. 9. A simulated observed image (blurred and noisy rendering) of the asteroid 433 Eros surface observed during the NEAR mission (3D model from the NASA Planetary Data System) [15]. Uniform albedo, Lambertian reflectance model, ambient/direct light ratio 10%, Gaussian blur (width 2 pixels) and Gaussian noise (variance 1% of the max. image intensity).

If we make the assumption that the global PSF can be well approximated by a piecewise constant function, made of linear combinations of the pixel PSF, then steps 2 and 3 can be swapped, and the convolution can be performed by a discrete filter denoted H, after performing the pixel integration as explained in the previous sections. Then, we simply add a discrete convolution step after the rendering, denoted by $I \star H$. The proposed rendering technique does not produce aliasing artifacts since it simulates the image formation process (most fast rendering algorithms produce aliased edges, since they rasterize triangles without performing any pixel integration).

The deterministic image formation, including both rendering and degradation by blur, can be summarized as follows:

- **Project** the surface vertices onto the image plane;
- **Determine the visible areas** of each triangle, for each pixel of the image (paragraph C.2);
- **Compute the shadows** for each triangle (paragraph C.3);
- **Compute the irradiance** for each visible triangle by using a reflectance model (Eq. (12));
- **Form the intensity** I for each pixel by combining visible areas and irradiance (Eqs. (9)-(10));
- **Blur the image** by convolution with a discrete PSF.

So far, we have only described the deterministic part of the image formation. The intensity measure in the camera sensor is a random process, because of the pixel noise (mainly due to photon, readout and thermal noise). We assume it can be modeled by a stationary white Gaussian noise of variance σ_p^2.

This enables us to write the conditional density of an observed image X, given the rendered intensity I. This density is also the likelihood of the parameters $(\mathbf{S}, \mathbf{\Theta})$:

$$P\big(X \,|\, I(\mathbf{S}, \mathbf{\Theta})\big) \propto \exp\left(- \sum_p \frac{\big(X - H \star I(\mathbf{S}, \mathbf{\Theta})\big)_p^2}{2\sigma_p^2} \right) \quad (14)$$

The hierarchy of the variables is shown in Fig. 1: each arrow represents a conditional density, and each leaf node a density encoding the prior knowledge about the related parameter. Thus we have the full joint density where W^{-1} denotes an inverse wavelet transform and the Dirac distributions δ account for deterministic relations between variables:

$$P(X, \mathbf{S}, \mathbf{\Theta}) = P(\mathbf{S}) P(\mathbf{\Theta}) P(X \,|\, I(\mathbf{S}, \mathbf{\Theta})) \quad (15)$$

$$P(\mathbf{S}) = P(q) P(\lambda) P(\gamma) P(\mathbf{w}^{\perp} \,|\, q, \lambda) P(\mathbf{w}^{\parallel} \,|\, \gamma) \delta(\mathbf{v} - \mathrm{W}^{-1}\mathbf{w})$$
$$\times P(m) P(\kappa \,|\, m) P(\varepsilon \,|\, m) P(\omega \,|\, \varepsilon) \delta(\tilde{\rho} - \mu(\mathrm{W}^{-1}\omega)) \mu'(\mathrm{W}^{-1}\omega)$$

All the densities involved in the equation above have been defined in Sections II.A and II.B. We give an example of simulated observed image in Fig. 9 for a known surface $\mathbf{S}$ and parameters $\mathbf{\Theta}$. We show simulations from the geometry model in Fig. 5 (assuming uniform albedo).

III. Potential applications and challenges

In many cases, computer vision can be seen as the inversion of a generative forward model. When such a model is probabilistic, a natural way of performing the inversion is via Bayesian inference [6]. Basically, it consists of computing a posterior density of the variables of interest, which is proportional to the joint density defined by the generative model. In general the full density is difficult to compute, and one prefers to estimate its maximum, or its mean. Whenever possible it is also useful to estimate the covariance matrix of the variables, since it represents the uncertainty on these variables.

The model presented here can have multiple applications: we can try to estimate the surface geometry, the albedo map, the reflectance map, the scattering properties and the fractal dimension of the surface, etc. We can also estimate the observation parameters to perform accurate camera calibration, PSF estimation, light calibration, etc. Estimating the reflectance map m consists of performing albedo classification. It is very important to understand that the classification should be performed on a physically meaningful terrain reflectance, not on image intensities which are the product of both reflectance and shading. The proposed model should help carry out such a classification since it clearly separates these two quantities.

A. Surface recovery from multiple images

Surface recovery consists of inferring the 3D surface model and the reflectance map from a set of images. As seen from the generative model the complex interplay between surface geometry and reflectance maps cannot easily be inverted. There does not exist a unique relation between an observed image and the underlying 3D object. However, using multiple images helps constrain the solution to the inverse problem. Moreover, the use of priors such as the ones we describe in this paper further constrains the solution, acting like a regularization process. Then surface inference becomes possible, as preliminary results have shown.

By restricting the observation parameters (camera parameters and light direction) to avoid shadows and occlusions and the model to a height field, we have shown that even when using a simplified version of the accurate renderer described in this paper, *accurate 3D reconstruction is possible from both simulated [23] and real data*. We have also assumed a Lambertian scattering model. A conjugate gradient algorithm

was used to maximize the posterior density given all the observed images, with derivatives computed as explained in paragraph II.C.4.

A physical model of the Duckwater, Nevada, area was constructed from the USGS digital elevation map. A CMOS panchromatic camera was used to image this model in sunlight (see Fig. 10). Camera pose and internal parameters were determined using the background checkerboard, and the sun angle was measured using a sundial. The inference started with a level surface, and converged to an estimate that is close to the original model (see Fig. 11): we obtained a maximum error less than 15mm with a 2m distance between camera and model. No existing stereo reconstruction method gave acceptable results in this case.

We have done another experiment with a spatially variable albedo, by painting the same physical model mentioned above (see Fig. 12). The same inference procedure has been used, but this time the albedo was allowed to vary. The inferred surface geometry shows RMS errors between 1 and 2mm, and maximum errors usually less than 10mm, which is better than in the constant albedo case. The results are shown in Fig. 13. The albedos look acceptable but their precision can not be quantified, since they have been added by hand (there is no ground truth).

Having a textured surface obviously helps reconstruct the 3D geometry. However, we have noticed some interaction between albedo and geometry, where abrupt albedo differences generate false slopes. To a lesser degree, the same problem occurs with extended albedo differences: smooth albedo variation generates shallow slopes in the heights.

We have demonstrated the feasibility, and the reduced computational complexity, of the posterior density optimization using intensity derivatives w.r.t. model parameters, in the case of height fields. By using wavelets on subdivided meshes as explained in Section II.A, it should be possible to infer objects of arbitrary topology such as entire planets, or asteroids. We have to investigate various ways of performing the

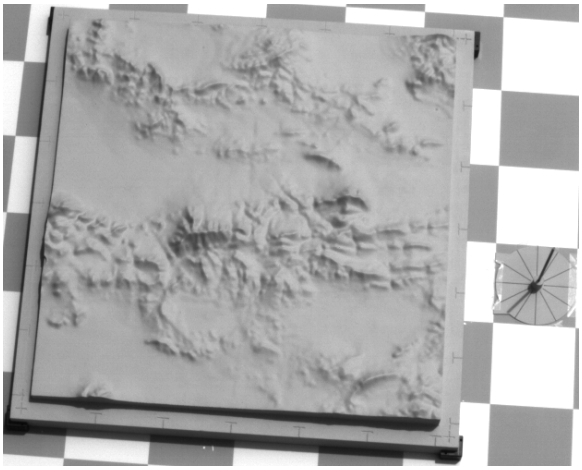

Fig. 10. One of the 16 observed images in our constant albedo experiment, showing the checkerboard and the sun dial used for sun calibration.

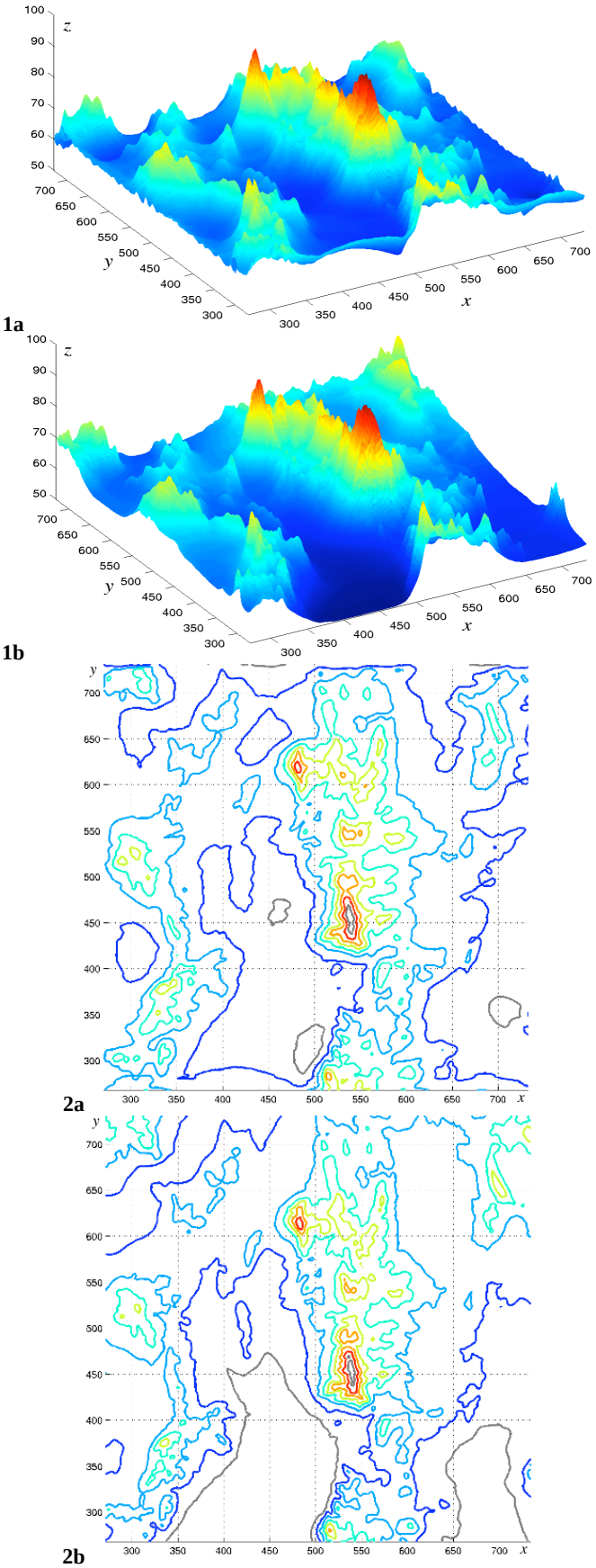

Fig. 11. **a**: estimated geometry model; **b**: original topography. The units are millimeters. **1**: 3D view of the height field $z = f(x, y)$; **2**: contour plot $z =$ const, with contours every 10mm.

optimization, for instance allowing the vertices to move in any direction, or constraining them to move along the local surface normal. When working in arbitrary topology, the initial mesh has to be deformed to fit the data. There are multiple solutions corresponding to various ways of arranging sampling points on the same surface. The mesh regularity prior described in this paper needs to be added to facilitate the optimization and improve the sampling regularity of the surface.

The generic model-based vision approach presented here avoids most of the shortcomings of existing methods in surface recovery, such as shape from shading [24] and shape from stereo [25]. The former is difficult to apply when the albedo is spatially variable, while the latter usually produces a sparse point set as a surface estimate. We can reconstruct continuous surfaces from multiple images, using different viewpoints and various lighting conditions.

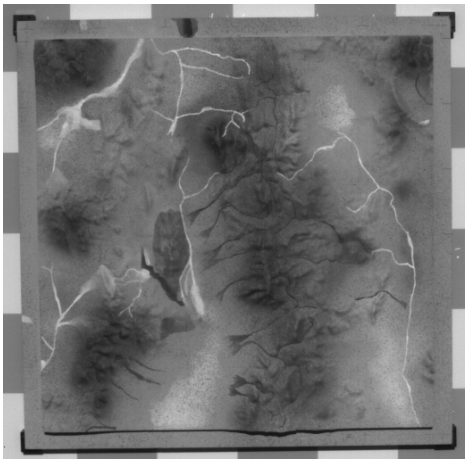

Fig. 12. One of the 8 observed images in our variable albedo experiment.

B. *Super-resolution*

Nothing prevents us from increasing the model resolution arbitrarily, thus achieving so-called *super-resolution*. However, there are some practical limitations such as the cutoff frequency of the optical system, and the limited amount of (noisy) data. We need to make certain that the design of the generative model does not bring any further shortcomings.

First, when surface triangle projections on the image plane are smaller than the pixel, which happens with increased surface resolution, computing accurate intensities in the object space is essential. Classical algorithms in computer graphics can not be used, because they perform image-based computations which are too approximate in this case. That is why we insisted on building an accurate rendering algorithm.

Second, the blur model shall preserve the spatial information of the high-resolution surface. For instance, a small projected triangle entirely contained in a pixel should produce a slightly different rendering when moved within the pixel. This is difficult with the current integration scheme which computes the area of visibility polygons; we believe it would be made possible by also computing the first order moments, because they are sensitive to the polygon location, even with small

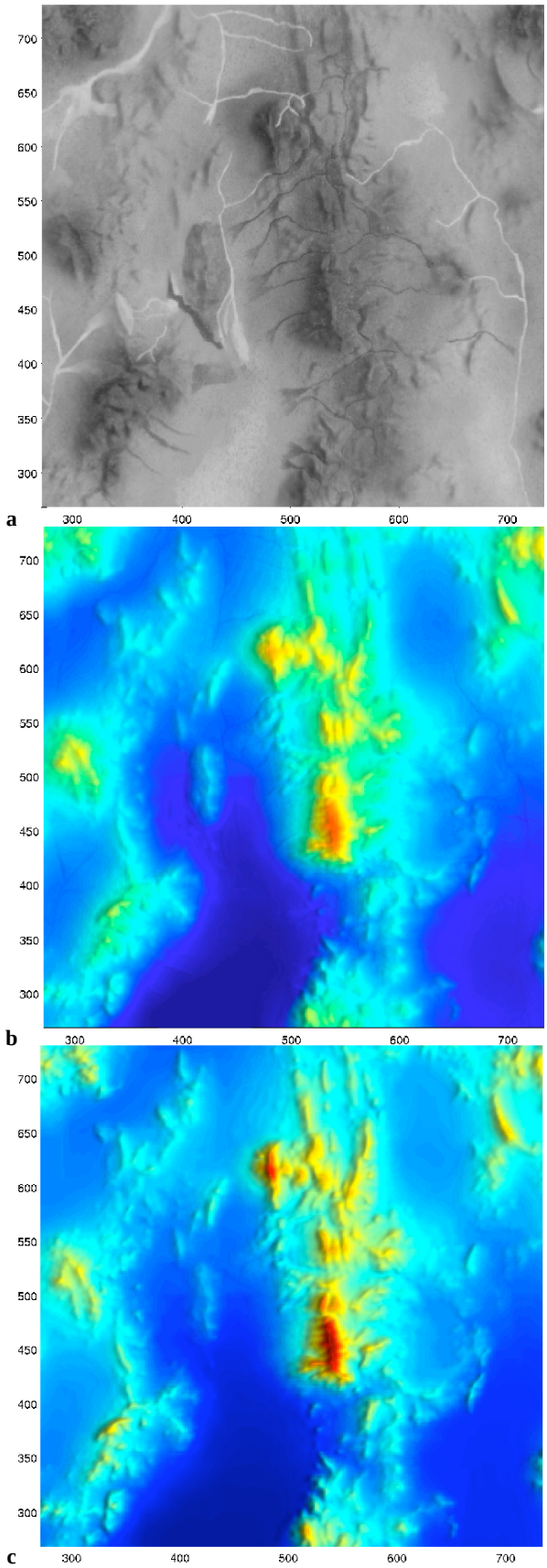

Fig. 13. **a**: inferred albedo field (black=0, white=1); **b**: inferred geometry model; **c**: original topography. The units are millimeters. The topography was rendered using Matlab, with the same color maps and limits, emphasized by a directed light.

triangles. A continuous displacement of tiny triangles would then result in a continuous variation of the rendered intensity.

C. Data fusion

Using this unified approach that takes into account *albedo, geometry and lighting* through a 3D surface model and an imaging model, it should be possible to use all the existing observations from a scene to build a single model that contains (almost) all the information present in this data. The observations include multiple images, with both geometric and shading content, but other modes (such as existing elevation models or odometry) could be included. Efficient *data fusion* [26] could be performed by building up a single model from multiple data sources. By computing the parameter uncertainties, it is possible to perform recursive updates of the model using a method such as the Kalman filter [27], instead of the estimation from all the data sources simultaneously which may be intractable with large volumes of data.

IV. Conclusion

We have proposed a detailed generative model to explain how both a complex natural surface and a realistic image of it is formed. When inverted within a Bayesian framework, this model leads to numerous applications in computer vision, such as surface recovery and camera calibration. It also has the potential to perform data fusion from large data sets, and to estimate reflectance and roughness properties of real surfaces.

We have demonstrated the feasibility of the proposed approach by reconstructing a 3D object from real images, using a simplified model. These results are promising, and we believe that more complex cases (such as occlusions, shadows, super-resolved surface and non-Lambertian reflectors) could be addressed by using a more accurate forward model such as the one we describe in this paper.

References

[1] A. P. Pentland, "Fractal-based description of natural scenes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, pp. 661–675, 1984.

[2] G. Burton and I. Moorhead, "Color and spatial structure in natural scenes," *Applied Optics*, vol. 26, pp. 157–170, 1987.

[3] K. Seidel and M. Datcu, "Fusion of real and synthetic images for remote sensing scene understanding," *FRACTAL'93, Kingston, UK*, 1993.

[4] D. Ruderman, "Origins of scaling in natural images," *Vision research*, vol. 37, no. 23, pp. 3385–3398, 1997.

[5] G. Huang and D. Mumford, "Statistics for natural images and models," *IEEE Proc. of CVPR*, pp. 541–547, 1999.

[6] J. M. Bernardo and A. F. M. Smith, *Bayesian Theory*. John Wiley & Sons, Inc., 1994.

[7] P. Kube and A. Pentland, "On the imaging of fractal surfaces," *IEEE Trans. on PAMI*, vol. 10, no. 5, 1988.

[8] B. Mandelbrot and J. V. Ness, "Fractional brownian motion, fractional noises and applications," *SIAM review*, vol. 10, no. 4, pp. 422–437, 1968.

[9] P. Schröder and W. Sweldens, "Spherical wavelets: efficiently representing functions on the sphere," *Computer Graphics*, vol. 29, no. Annual Conference Series, pp. 161–172, 1995.

[10] N. Dyn, J. Gregory, and D. Levin, "A butterfly subdivision scheme for surface interpolation with tension control," *ACM Trans. Graph.*, vol. 9, pp. 160–169, 1990.

[11] D. Zorin, P. Schröder, and W. Sweldens, "Interpolating subdivision for meshes with arbitrary topology," *Computer Graphics*, vol. 30, no. Annual Conference Series, pp. 189–192, 1996.

[12] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," *SIAM Journal on Mathematical Analysis*, vol. 29, no. 2, pp. 511–546, 1998.

[13] I. Guskov, W. Sweldens, and P. Schröder, "Multiresolution signal processing for meshes," in *Siggraph 1999, Computer Graphics Proceedings*, A. Rockwood, Ed. Los Angeles: Addison Wesley Longman, 1999, pp. 325–334.

[14] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," *Computer Graphics*, vol. 29, no. Annual Conference Series, pp. 173–182, 1995.

[15] M. Z. et al., "The shape of 433 Eros from the NEAR-Shoemaker laser rangefinder," *Science*, vol. 289, pp. 2097–2100, 2000.

[16] G. Wornell, *Signal processing with fractals: a wavelet-based approach*, ser. Signal processing series. Prentice Hall, 1995.

[17] E. Simoncelli, "Bayesian denoising of visual images in the wavelet domain," in *Bayesian inference in wavelet-based methods*, P. Muller and B. Vidakovic, Eds. Springer Verlag, 1999.

[18] A. Jalobeanu, L. Blanc-Féraud, and J. Zerubia, "Natural image modeling using complex wavelets," in *Wavelets X, SPIE Symposium*, San Diego, CA, Aug. 2003.

[19] B.-T. Phong, "Illumination for computer generated images," *Communications of the ACM*, vol. 18, no. 6, 1975.

[20] S. K. Nayar and M. Oren, "Generalization of the Lambertian model and implications for machine vision," *International Journal of Computer Vision*, vol. 14, 1995.

[21] K. Weiler and P. Atherton, "Hidden surface removal using polygon area sorting," in *Proc. of the 4th annual conf. on Computer graphics and interactive techniques*. ACM Press, 1977.

[22] E. Catmull, "A hidden-surface algorithm with anti-aliasing," in *Proc. of the 5th annual conf. on Computer graphics and interactive techniques*. ACM Press, 1978.

[23] V. Smelyanskiy, P. Cheeseman, D. Maluf, and R. Morris, "Bayesian super-resolved surface reconstruction from images," *Proc. of CVPR*, June 2000.

[24] B. Horn and M. Brooks, *Shape from Shading*. MIT Press, 1989.

[25] R. Deriche, Z. Zhang, Q.-T. Luong, and O. Faugeras, "Robust recovery of the epipolar geometry for an uncalibrated stereo rig," *Lecture Notes in Computer Science*, vol. 800, pp. 567–576, 1994.

[26] J. K. Aggarwal, Ed., *Multisensor Fusion for Computer Vision*. NATO ASI Series F, Vol. 99, Springer-Verlag, 1989.

[27] D. Maluf, P. Cheeseman, V. Smelyanskiy, F. Kuehnel, and R. Morris, "The 3D recognition, generation, fusion, update and refinement (RG4) concept," *Proc. of the Sixth International Symposium on Artificial Intelligence and Robotics and Automation in Space*, June 2001.